Plug–in Grid: A Fully Virtualized Grid Cluster

Release 0.00

Marko Niinimaki[1], Xin Zhou[1], Adrien Depeursinge[1], and Henning Müller[1,2]

July 27, 2009

[1]Medical Informatics, University Hospitals and University of Geneva, Switzerland
[2]Business Information Systems, University of Applied Sciences Western Switzerland, Sierre, Switzerland

Abstract

Medical image processing is known as a computationally demanding and data intensive field. For parallelizing the processing of image data, Grid computing systems and methods have been successfully applied. However, installing and maintaining Grid clusters is a demanding (and often non–rewarding) task for researchers. In this paper, we describe a Grid system that can be deployed completely within virtual machines on standard PC's. The system consists of a Grid/Cluster server and a large number of computing nodes. We discuss its features and demonstrate its performance with real–life tests using several medical imaging applications. Impact of the virtual machine with the computing nodes on the desktop speed are measured and compared on various computers and in various scenarios.

Latest version available at the Insight Journal [`http://hdl.handle.net/1926/1338`]
Distributed under Creative Commons Attribution License

Contents

1 Introduction

Modern hospitals produce ever–increasing quantities of data, much of it in the form of digital images, including tomography slices [15]. Medical doctors, analysts and researchers struggle with such an influx of data, particularly in hospitals where there are no dedicated computing resources for research, which is most frequently the case. Cluster computing and combining clusters by so–called Grid middlewares have potential in such environments, especially if the task at hand can be parallelized easily (see [18, 14]).

Harnessing the power of an organization's desktop PCs as a cluster is an intriguing concept, and implemented often using the Condor cluster software [10] in projects such as Greedy [19]. However, a hospital environment often has strict policies that prevent ad–hoc software installations on computers. To overcome this difficulty, projects like Grid Appliance [20] and CoreGrid [11] use a Virtual Machine (VM) within which the software is installed (also solving the problem that the process running in the virtual machine does not have access to the potentially confidential data on the client). As emphasized by Figueiredo et al. [5], this design isolates the application environment from the host PC, improving stability and security. The cluster software is, likewise, run inside the Virtual Machine. The KnowARC[1] project's GridFactory [17] presents a different design, where a cluster software starts Virtual Machine instances in computers, installs software and manages jobs in them. At the MedGIFT[2] research group of the Geneva University Hospitals (HUG), we have tested both approaches using a medical image indexing task as a case [13, 15, 17].

The design presented in this paper follows the same principles as in [15] — namely:

- A cluster of standard hospital PCs is running identical Virtual Machines, containing a compact Linux operating system with tools, applications, and a Condor worker node software (Software distribution is fully automatic via the Microsoft Active Directory–based hospital solution).
- A central server in the hospital runs a Grid middleware, gets jobs from users, sends them to worker nodes (using Condor), receives and stores the results. The results can then be retrieved by the user.

Contrary to our previous implementation, in the scenario described in this paper, we have isolated the Grid server as well, and it is run in a virtual machine. The virtual machine physically resides on an external hard disk, making the system fully portable. Moreover, due to the simple design of the Grid node VM image, it can be copied to any standard hospital PC, thus adding another node into the Grid resources. Another difference to the previous paper are the details tests of the impact of the virtual machine on the desktop performance of the host machine.

The most obvious benefit of this system is that with it the researchers in the hospitals can run analyses that would be impossible without a Grid/Cluster system (see Section Applications). The benefit for using a Grid layer on top of Condor allows the users to run their analysis in another ARC (Advanced Resource Connector) Grid outside of the hospital, if needed. The Taverna workflow engine [16] with its ARC plug–in [8] provides the users with a generic graphical interface for designing analysis tasks, without the need for detailed knowledge of the underlying Grid system.

[1] `http://www.knowarc.eu/`
[2] `http://www.sim.hcuge.ch/medgift/`

The rest of the paper is organized as follows. In Section 2 the characteristics of the Grid nodes, the central server and the environment are explained. Test scenarios for running performance and impact tests are described. In Section 3 we present the results of these tests. Section 4 discusses applications that utilise our Grid. Finally, Section 5 contains a short summary and discussion.

2 Methods

This section describes the environment and systems that are the basis for the work described in this paper.

2.1 Existing hospital environment

Like many medical institutions, the HUG do not have a dedicated research computing infrastructure. On the other hand, a very large number of desktop PC's is available, and the renewal cycle for these PCs is 4 to 5 years [21]. Currently (early 2009), around 6,000 computers are available, with the slowest ones containing 1 GB of main memory and a single Pentium IV 2.8 GHz CPU. By mid–2009 around 5,000 of the PC's will have at least 2 GB of main memory and at a minimum 2.6 GHz dual core CPUs. The dual core CPU supports virtualization in hardware, as well [7], leading to a much better performance.

As described in [15], 20 old hospital PCs in a seminar room were made available for us to create an intra–hospital Grid. These computers became computing nodes by virtualization. In addition to these computers, several desktop PC's of the researchers are used as additional nodes in the same way. These are the new generation dual–core PCs.

2.2 Grid setup at the HUG

Condor is used as the cluster software, NorduGrid's Advanced Resource Connector (ARC) [4] 0.6.5 is used as the Grid middleware for job submission and management.

A Debian Linux based Virtual Machine image, called Grid node, and Virtual Machine Player (VMPlayer[3]) are distributed in the hospital to a set of standard PC's running Windows XP. VMPlayer starts the Debian image when the PC is started, though the users of the PC can turn it off if they need more CPU power or memory for a particular task.

The Grid nodes are configured to use 350 MB of memory and a maximum of 2 GB of disk space. They receive an IP (Internet Protocol) address by DHCP (Dynamic Host Configuration Protocol) from the hospital's server.

The Grid nodes communicate with a (pre–configured) central server that runs the *Condor collector* process for sending jobs to and receiving results from Grid nodes. The central server also runs the ARC server. The ARC server manages Grid jobs by getting job descriptions from the users, submitting them to Condor, receiving the results, and storing the results to be retrieved by the user.

Figure 1 shows the overall structure of the infrastructure implemented in the hospitals. A central node stored on an external hard disk as virtual machine is controlling the working nodes that are in different computing rooms of the hospitals, also in virtual machines running Linux on standard desktop PCs.

[3] `http://www.vmware.com/products/player/`

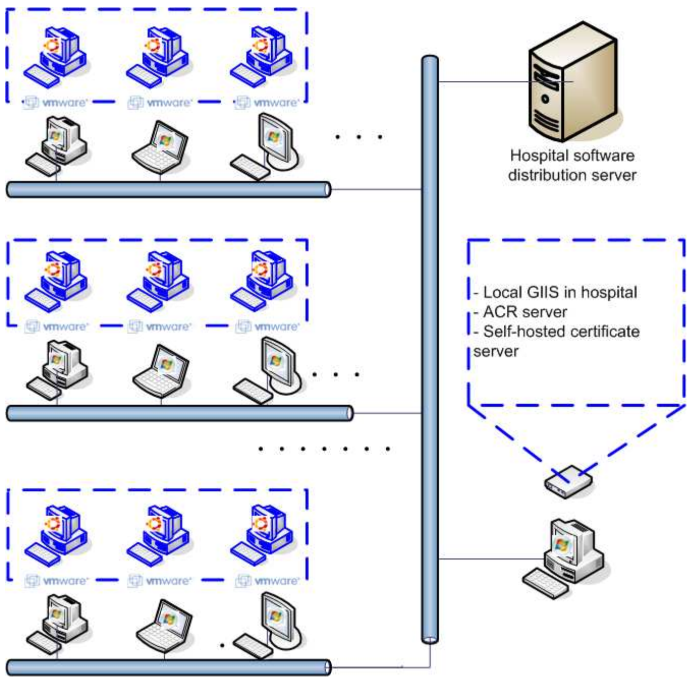
vmware
Hospital software distribution server
- Local GIIS in hospital
- ACR server
- Self-hosted certificate server

Figure 1: Overview of the infrastructure implemented for a fully virtual Grid, including the central node stored as virtual machine on an external hard disk.

2.3 Evaluating a fully virtual Grid

To test the system's performance and its impact to the computers, we have performed the following tests. It should be noted that here we test single tasks and their impact to individual computers. Section 4, however, discusses the performance of the Grid system as a whole.

- Performance (speed) test: compare the execution time of an image analysis job in a virtual machine with its execution time without virtualization.
- Impact tests on the desktop computers running the virtual machines:
  - Run a benchmark on the host computer (1) when there is no virtual machine running (2) when the virtual machine is running image analysis tasks. The benchmark program (with the virtual machine in case 2) is the only application running in the host computer during this test.
  - Compare the startup times of some typical applications in the host computer (1) when there is no virtual machine running, (2) when the virtual machine is running but idle, and (3) when the virtual machine is running image analysis at full capacity.

These tests were requested by the computer service of the hospitals to estimate the impact of the system on desktop users. Moreover, they allow estimation what kind of desktops could potentially be used for our internal Grid. We also measure network bandwidth and ways to limit this. The goal of this was also to evaluate the impact of such a system on the entire network.

	Internet Explorer 7	Microsoft Office 2003
VM not running	2.7	2.8
VM running but idle	3.2	3.3
VM running analysis	5.8	8.1

Table 1: Startup times of some of the hospital's frequently used applications on a single–CPU PC.

3 Results

This section details the results of the tests with the hospital Grid and its impact.

3.1 Performance measurements

As a simple but representative example, we have measured the execution time of one task in a virtual machine and in a native Linux operating system (installed on an identical computer). The task contains unpacking a *tar* file containing 100 images, compiling a feature extraction program, running the feature extraction for each of the images and combining the results in a *tar* file. This task is a very typical task for image retrieval that is extremely simple to parallelize.

On a single–CPU Pentium IV computer (old desktops used for our Grid) with a native Linux operating system this task takes 3 min 13 seconds, and on a Virtual Machine in the same computer 4 min 15 seconds. Interestingly, on the dual–core CPU computer (researchers personal machines) the execution time is almost identical in native Linux and in a virtual machine (1 min 43 seconds vs. 1 min 45 seconds).

3.2 Impact measurements

A criterion for the usability of the system in the hospitals has been that standard desktop PCs provided for hospital administrative staff and nurses can run Grid nodes so that the users do not notice a large detrimental effect on the performance.

With a single CPU system (our test Grid), this was not the case. Our measurements with the NovaBench[4] benchmark software indicated that the performance of a five–year–old standard hospital PC without the Virtual Machine was 35 MFLOPS (Million FLoating Point Operations per Second), 6.4 M integer operations/s. With the Grid node running but idle, the figures were 34 MFLOPS and 6.3 M integer operations/s. When the Grid node was running image analysis, the performance figures were 18 MFLOPS, 2.3 M integer operations/s [15]. In practice this meant that the startup time of the default WWW browser (Internet Explorer 7) degraded from 2.7 seconds to 5.8 seconds. The effect was even more noticeable in the startup times of Office applications, though hardly noticeable in text processing and spreadsheet usage once the applications were running. Table 1 shows the application startup times on a single CPU system.

With the new generation of dual–core processor PCs (90% of all PCs by mid 2009), Grid nodes run very well in the virtual machine, and they have only a very limited effect on the perceived performance of the PC, since they use only one of the processor cores. The startup times of the most commonly used application software packages are shown in Table 2. The respective figures with the NovaBench2 benchmark were 62 MFLOPS and 34 M integer operations per second when the VM was not running, 60 MFLOPS and 31 M integer operation when the VM was running but idle, and 56 MFLOPS, 28 M integer operations per second

[4] `http://novabench.com/`

	Internet Explorer 7	Microsoft Office 2003	Microsoft Office 2007
VM not running	2.7	2.4	4.2
VM running but idle	2.7	2.5	4.5
VM running analysis	2.8	3.0	6.1

Table 2: Startup times of some of the hospital's commonly used applications on a dual–CPU PC.

when Grid node was running an analysis job. In reality these differences are not really noticeable for an end user.

The improved performance of desktops has given us the possibility to run the central server inside a virtual machine as well. ARC software is relatively lightweight; an analysis of ARC 0.6.5's memory usage indicates that it consumes about 90 MB of the VM memory in operation (grid infosystem: 20 MB, gridftp 11.5 MB, grid manager 53 MB, scripts communicating with Condor 5 MB). However, *staging* jobs (copying data to and from Condor and making it available for the user) can take large amounts of disk space. Thus, we have prepared the Virtual Machine so that it can use disk space dynamically without limitations. The memory usage is set to 1 GB (50% of the dual–core CPU PC's physical memory). An inexpensive 1 TB external hard disk is used for storage and is also hosting the virtual machine itself.

Grid middlewares usually rely on a certificate–based authentication and authorization framework [6]. For a Grid server, the name of the computer (the fully–qualified hostname) must match with the subject of the computer's certificate — otherwise the communication with this computer is rejected by the Grid client software. In the hospitals' DHCP setup, a specific Virtual Machine is always given the same IP address. Naturally, the names of the computers are determined by the IP addresses. Therefore, we can physically move the external hard disk, containing the Virtual Machine image, to another PC if needed. Thus, we are creating a fully virtual intra–hospital Grid system, where the nodes as well as the controlling nodes can be moved on the standard desktops easily and quickly.

4 Applications

In this section, we present the medical applications that have utilized our hospital Grid and profited from the additionally available computing power. The current applications are limited to medical imaging but other applications such as natural language processing or data mining can easily be adapted to this scenario.

4.1 GIFT feature extraction

The GNU Image Finding Tool (GIFT[5]) has been used as a benchmark for computation in papers by the MedGIFT group as a scenario for content–based medical image retrieval [13]. The 50 0000 source images are from the ImageCLEFmed 2007 collection ([3][6]). For the computation, the collection is divided into packages containing a fixed number of source images, and the image feature extraction software taken from the GIFT software (GNU Image FindingTool). Previously, a 4–CPU local system was used for analysis, with an execution time of 709 minutes. By using the cluster of rather old desktop computers (much slower CPUs), an execution time of 240 minutes was achieved [15].

[5] `http://www.gnu.org/software/gift/`
[6] `http://www.imageclef.org/`

4.2 Lung tissue analysis with the TALISMAN software

TALISMAN (Texture Analysis of Lung ImageS for Medical diagnosis AssistaNce) is Java software with a front-end GUI and back-ends for distributed analysis [2]. Two main tasks were griddified for TALISMAN: feature extraction using wavelets, and image classification using Support Vector Machines. The source images that are analysed are high resolution computed tomography (HRCT) images of the chest.

Finding features that reveal lung illnesses is computationally very demanding and thus a distributed solution is much needed. In the analysis, the whole wavelet decomposition (convolutions) and feature calculation (mean and variance of the wavelet coefficients as well as grey level-histograms) were run on the Grid. The features are currently extracted from regions of interest in the images only. The region of interest was defined manually before the analysis (in our more recent version this is automatic).

In the application, we have done feature extraction for a series of images containing 30 slices (on average). The dimensions of each slice are 512x512 pixels. 58 series were used in the test.

The execution time of analysis in a single computer was more than 6 hours. By using ARC and Condor nodes in our cluster, the execution time was cut to 109 minutes.

Eventually, the features should be extracted on “per pixel” basis, creating a much larger need for computing power, currently not even attempted but possible through the Grid.

4.3 SIFT workflows with TAVERNA and ARC

The Scale–Invariant Feature Transform (SIFT) method is often used for feature extraction from medical images as well as for more general stock photography. A workflow process for this task was designed using the Taverna workflow engine [16]. Taverna communicates with ARC using an integration plug–in from the KnowARC project[7] [8]. The implementation was adapted from ImageJ’s SIFT plug–in (see [1]). The source images were selected from the ImageCLEFmed 2007 collection. The running times in the cluster varied between 3 to 5 hours (meaning that running the whole analysis as one process would have taken more than 1 week). As SIFT features can have a large variety of parameters, which can largely determine performance, it is important for us to perform this systematic testing. By being able to test new parameters within hours instead of days we have many more options than beforehand. Potential end users of this system are surgeons who contacted us regarding a project on fracture image retrieval. This technology aims at being applied on fracture image retrieval that is described in [12] for a first pilot application.

5 Conclusions and discussion

In this paper, we demonstrate the feasibility of an internal Grid in a hospital environment using standard hospital desktop PCs. The Grid system, including the server, is fully based on virtualization, and standard hospital PC’s are used as computing nodes. The benefit of this setup, compared to our previous one, is that is it very portable, does not require a specialized setup in any node, and its performance is still good.

As of now, the system can be best described as functional prototype with a small but active set of users. In order to present the system to larger user community, many political and technical problems would still need to solved, among them providing intuitive interfaces for non–programmers (emphasized e.g. in [9]).

To summarize, the Grid system works as follows:

[7] http://www.knowarc.eu/

- The building blocks of the system are the Condor batch system and NorduGrid ARC grid middleware.
- The Condor execution nodes (Grid nodes) are run on standard hospital PC's in a Virtual Machine image. Currently, a test bed of 20 computers, and several researcher PCs are running the Grid nodes. The images are identical and can be copied to additional PCs for expanding the cluster (a fully automatic solution exists using the standard hospital software distribution system based on Microsoft Active directory).
- The Grid server runs in a virtual machine with large disk space, for staging Grid jobs. Like the Grid nodes, the server can be moved to another location (another host PC) easily as only a single external harddisk needs to be moved from one computer to another one.

We present use case applications and performance measurements of the implemented solution. The impact of the Grid nodes on the performance of the host PC is measured by benchmarks and by startup times of popular applications. In modern dual–CPU host computers, the impact is generally not noticeable by the user. On five–year old desktop PCs on the other hand the performances degrades in an important manner particularly for application startup times.

Our Grid applications consist of parallel image processing tasks for three differing applications. A notable recent improvement for easing the creation of Grid–enabled applications is the use of the Taverna workflow engine in designing and running the tasks. This system allows for a graphical way of combining application blocks and does not require command–line based tools. Taverna's ARC plug–in enables the user to run the tasks in ARC–based Grids.

The experiences show that small Grids within medical institutions are possible and that virtualization techniques work well on new desktop computers. A Grid node running in a virtual machine on a user's desktop does barely slow the use of standard desktop applications. On the other hand, research applications can profit from the availability of more computing power allowing quicker tests of parameters and more complex solutions.

Acknowledgements

This work was partially supported by the EU 6th Framework Program in the context of the KnowARC project (IST 032691) and by the Swiss National Science Foundation (FNS) in the context of the Talisman–2 project (project 200020 118638).

References

[1] Wilhelm Burger and Mark J. Burge. *Digital Image Processing, An Algorithmic Introduction Using Java*. Springer, 2008. 4.3

[2] Adrien Depeursinge, Jimison Iavindrasana, Asmâa Hidki, Gilles Cohen, Antoine Geissbuhler, Alexandra Platon, Pierre-Alexandre Poletti, and Henning Müller. Comparative performance analysis of state–of–the–art classification algorithms applied to lung tissue categorization. *Journal of Digital Imaging*, 2009–to appear. 4.2

[3] Thomas Deselaers, Thomas M. Deserno, and Henning Müller. Automatic medical image annotation in ImageCLEF 2007: Overview, results, and discussion. *Pattern Recognition Letters*, 29(15):1988–1995, 2008. 4.1

[4] Mattias Ellert, Michael Grønager, Aleksandr Konstantinov, Balázs Kónya, J. Lindemann, I. Livenson, Jakob Langgaard Nielsen, Marko Niinimäki, Oxana Smirnova, and Anders Wäänänen. Advanced resource connector middleware for lightweight computational Grids. *Future Generation Computer Systems*, 23(2):219–240, 2007. 2.2

[5] R. Figueiredo, P. Dinda, and J. Fortes. A case for Grid computing on virtual machines. In *Proceedings International Conference on Distributed Computing Systems (ICDCS)*, pages 550–559, 2003. 1

[6] Ian Foster, Carl Kesselman, Gene Tsudik, and Steven Tuecke. A security architecture for computational grids. In *CCS '1998: Proceedings of the 5th ACM conference on Computer and communications security*, pages 83–92, San Francisco, California, United States, 1998. 3.2

[7] Intel. Intel virtualization technology. *Intel Technology Journal*, 10(3), 2006. 2.1

[8] Hajo N. Krabbenhöft, Steffen Möller, and Daniel Bayer. Integrating ARC Grid middleware with Taverna workflows. *Bioinformatics*, 24(9):1221–1222, March 2008. 1, 4.3

[9] D. Krefting, J. Bart, K. Beronov, O. Dzhimova, Falkner J., M. Hartung, Hoheisel A., T. A. Knoch, T. Lingner, Y. Mohammed, K. Peter, E. Rahm, U. Sax, D. Sommerfeld, T. Steinke, T. Tolsdorff, M. Vossberg, F. Viezens, and A. Weisbecker. Medigrid: Towards a user friendly secured grid infrastructure. *Future Generation Computer Systems*, 25:326–336, 2009. 5

[10] M. Litzkov, M. Livny, and M. Mutka. Condor — a hunter of idle workstations. In *Proceedings of the 8th international conference on distributed computing*, pages 104–111, San Jose, California, USA, June 1988. 1

[11] A. C. Marosi, P. Kacsuk, G. Fedak, and O. Lodygensky. Using virtualmachines in desktop grid clients for application sandboxing. Technical report, CoreGrid, 2008. 1

[12] Henning Müller, Phuong Anh Do Huang, Adrien Depeursinge, Pierre Hoffmeyer, Richard Stern, Christian Lovis, and Antoine Geissbuhler. Content–based image retrieval from a database of fracture images. In Steven C. Horii and Katherine P Andriole, editors, *Medical Imaging 2007: PACS and Imaging Informatics*, volume 6516 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, page 65160H, San Diego, CA, USA, March 2007. 4.3

[13] Henning Müller, Nicolas Michoux, David Bandon, and Antoine Geissbuhler. A review of content–based image retrieval systems in medicine – clinical benefits and future directions. *International Journal of Medical Informatics*, 73(1):1–23, 2004. 1, 4.1

[14] Henning Müller, Mikko Pitkanen, Xin Zhou, Adrien Depeursinge, Jimison Iavindrasana, and Antoine Geissbuhler. Knowarc: Enabling Grid networks for the biomedical research community. In *Healthgrid 2007*, pages 261–268, Geneva, Switzerland, April 2007. 1

[15] Marko Niinimäki, Xin Zhou, Adrien Depeursinge, Antoine Geissbuhler, and Henning Müller. Building a community grid for medical image analysis inside a hospital, a case study. In Silvia D. Olabarriaga, Diane Lingrand, and Johan Montagnat, editors, *Medical imaging on grids: achievements and perspectives (Grid Workshop at MICCAI 2008)*, pages 3–12, New York, USA, September 2008. 1, 2.1, 3.2, 4.1

[16] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and P. Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, 2004. 1, 4.3

[17] Frederik Orellana, Marko Niinimäki, Xin Zhou, Peter Rosendahl, Henning Müller, and Anders Wäänänen. Image analysis on gridfactory desktop grid. In *HealthGrid 2009*, Berlin, Germany, July 2009. 1

[18] J. M. Squyres, A. Lumsdaine, and R. L. Stevenson. A toolkit for parallel image processing. In *Proceedings of the SPIE Conference on Parallel and Distributed Methods for Image processing*, pages 69–80, San Diego, CA, July 1998. 1

[19] M. Thiemard and P. Jermini. Grid@EPFL — desktop grid using condor. In *EGEE–06*, Geneva, Switzerland, September 2006. 1

[20] D. I. Wolinsky, A. Agrawal, P. O. Boykin, J. R. Davis, A. Ganguly, V. Paramygin, Y. P. Sheng, and R. J. Figuereido. On the design of virtual machine sandboxes for distributed computing in wide-area overlays of virtual workstations. In *Workshop on Large–Scale and Volatile Desktop Grids (PCGrid)*, March 2007. 1

[21] Xin Zhou, Hajo Krabbenhöft, Marko Niinimäki, Adrien Depeursinge, Steffen Möller, and Henning Müller. An easy setup for parallel medical image processing: Using Taverna and ARC. In *Proceedings of HealthGrid 2009*, Berlin, Germany, 2009. 2.1

Latest version available at the Insight Journal [http://hdl.handle.net/1926/1338]
Distributed under Creative Commons Attribution License